**cd ..**

****





Desarrollo Front-End Web

**Grupo:** 2

**Nombre del grupo:**

DevelopersEC

**Integrantes:**

Marlon Cevallos

Michael Gudiño

Carla Iza

Steven Jaramillo

**Docente:**

Jorge Edison Lascano



**Grupo 2**

**DevelopersTeam2FashionStore**

**Git:** [**https://github.com/MarlonCevallos/DevelopersTeam2FashionStore.git**](https://github.com/MarlonCevallos/DevelopersTeam2FashionStore.git)

**Mongo Atlas: mongodb+srv://MarlonTeam2:2022@**[**cluster0.pvig5s9.mongodb.net/FashionStoreDB?retryWrites=true&w=majority**](http://cluster0.pvig5s9.mongodb.net/FashionStoreDB?retryWrites=true&w=majority)

**Video:**

[**https://youtu.be/pFs1pmzl2iQ**](https://youtu.be/pFs1pmzl2iQ)

| **Activity report** | |
| --- | --- |
| Marlon Cevallos | * Desing - Register Product * Desing - Products list * Desing - InfoClientPage |
| Michael Gudiño | * Desing - Register Provider * Desing - Providers list * Desing - AbstractPage |
| Carla Iza | * Desing - Register Customer * Desing - Customer list * Desing - ApiPublicUrl |
| Steven Jaramillo | * Desing - Login * Desing - Home page (session) * Design - PublicApi |

**1. Github Repository (Right Structure and collaboration) /1**

<https://github.com/MarlonCevallos/DevelopersTeam2FashionStore.git>

**2. Design (sketches) of the web pages /0.5**

**2)** **Public API Consumption**

**URI:** [**https://api.plos.org/search?q=title:github**](https://api.plos.org/search?q=title:github)

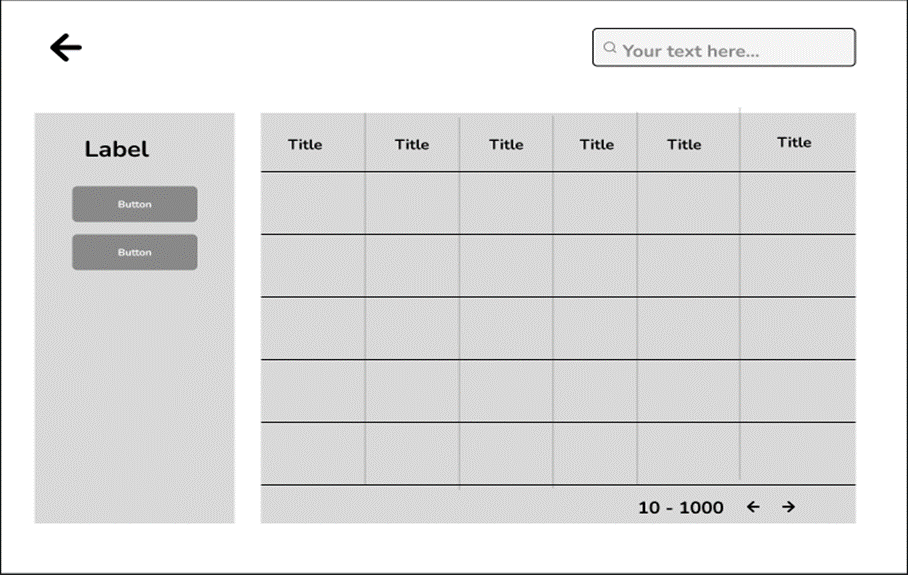
You have to implement one web page where the user will enter the search term. The user must select how they want to view the data. The developer is free to design the user interface of these five web pages, but you must use a framework to implement it later. The form (and its results) and the four pages are dynamically created with the retrieved data as follows:

* The developer is free to design the user interface of these five web pages.
* You must implement a web page where the user will enter the search term.
* This page shows the tabulated documents (does not include the abstract) of each one of the articles returned by URI 1, the titles of the table columns are: ID, journal, eissn, publication date, type\_of\_article, screen\_author, display\_of\_title , punctuation

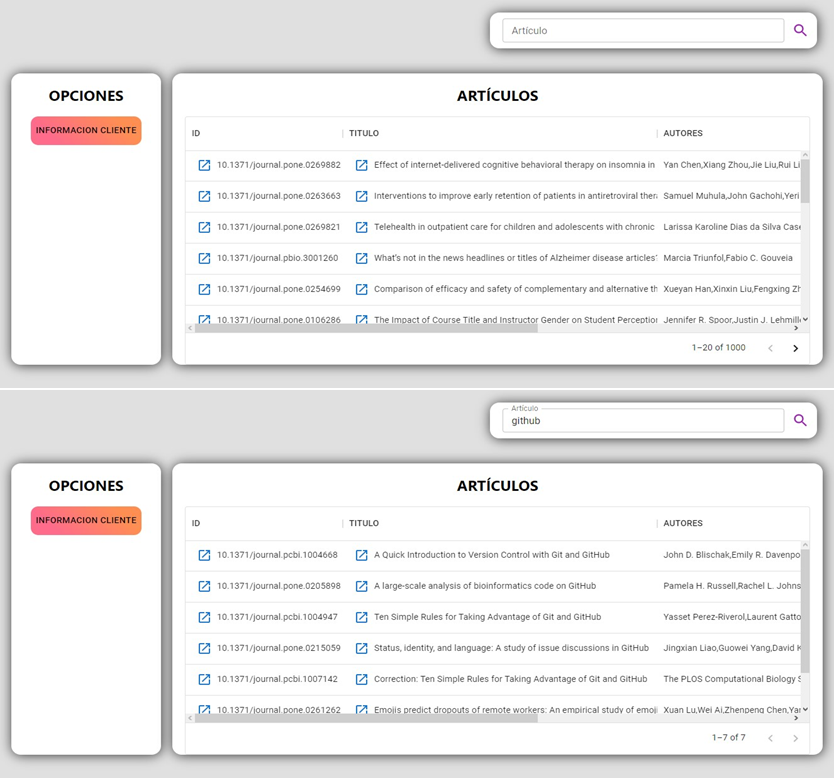
**This page shows the tabulated documents (do not include the abstract) of each of the articles returned by URI 1, the titles of the columns are:**

* id,
* journal,
* eissn,
* publication\_date,
* article\_type,
* author\_display,
* title\_display,
* score

**2.1 Design:**

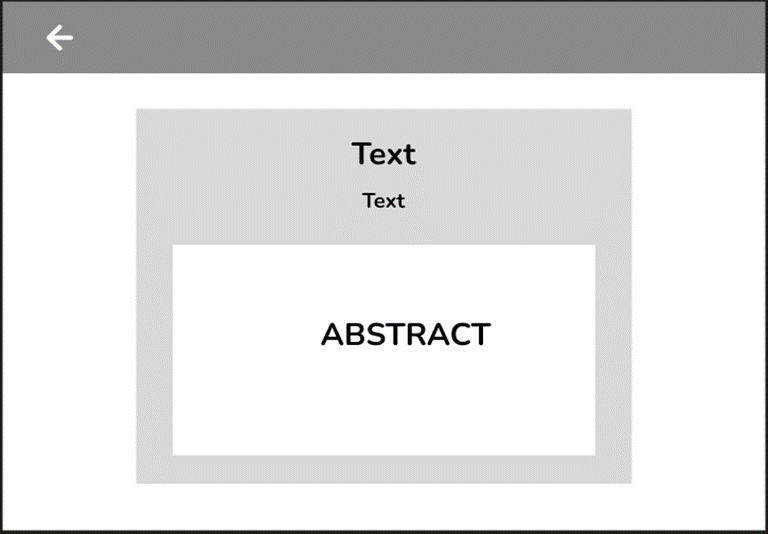


**2.1.1 Implementation:**

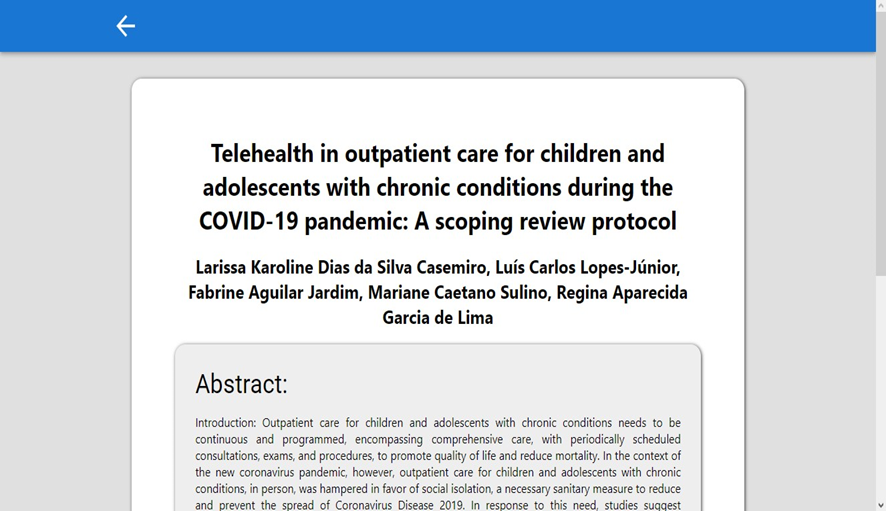
****

**On the previous page, for each title of the papers, program a link that displays the abstract for that paper on a new page as read from the URI.**

**2.2 Design:**

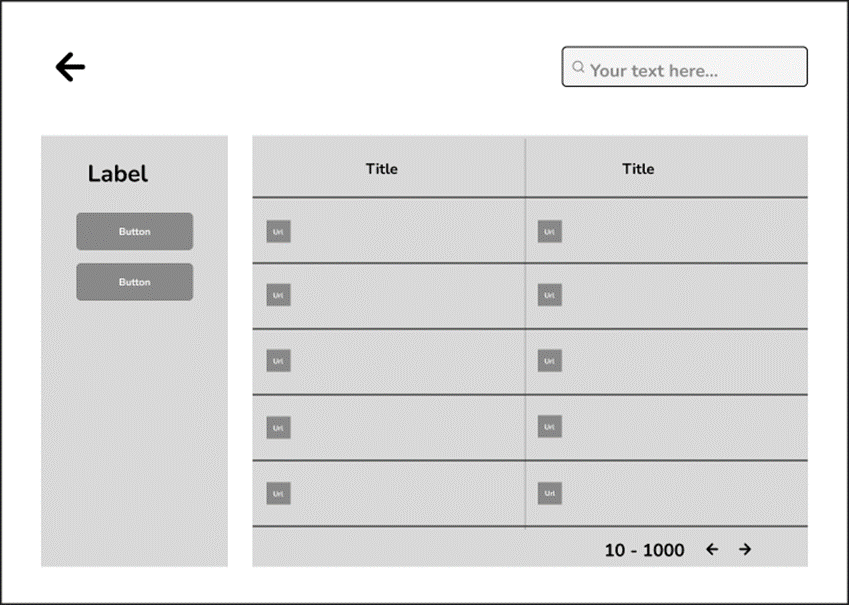
****

**2.2.1 Implementation:**

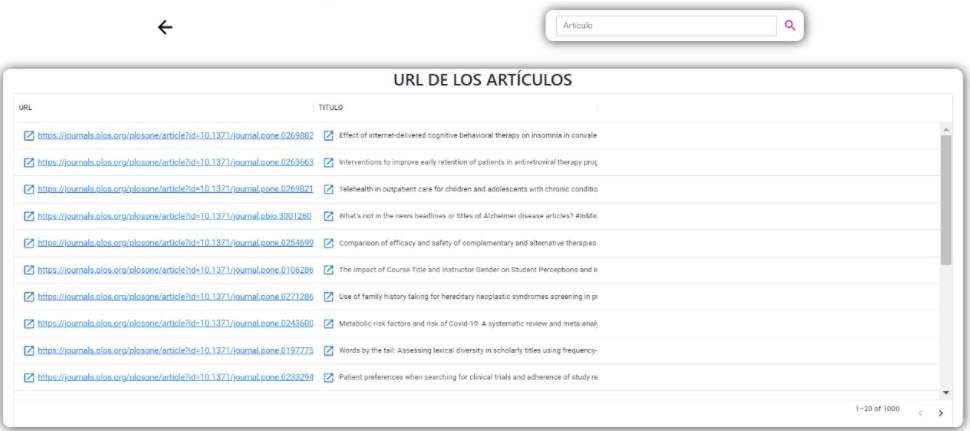
****

**A new page shows a table of every document identifier, with their respective id. From each id, a URL must be created so that the entire document can be viewed in a new tab (a new web browser tab), not in the same web application you are developing.**

**2.3 Design:**

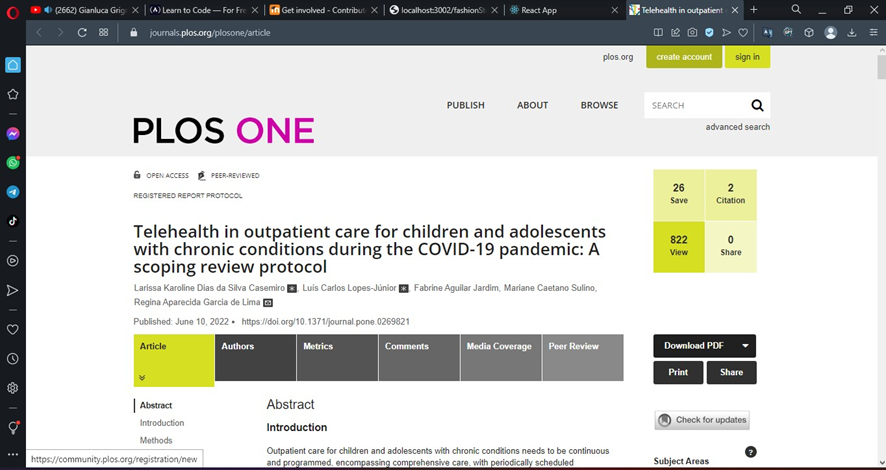
****

**2.3.1 Implementation:**

****

**2.3.2**

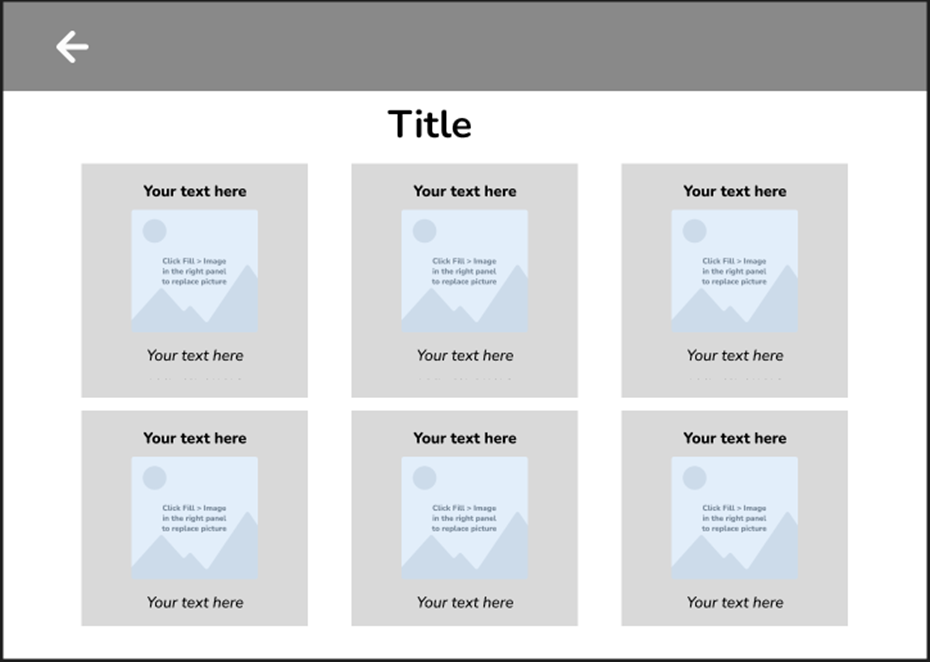
**New web browser tab:**

****

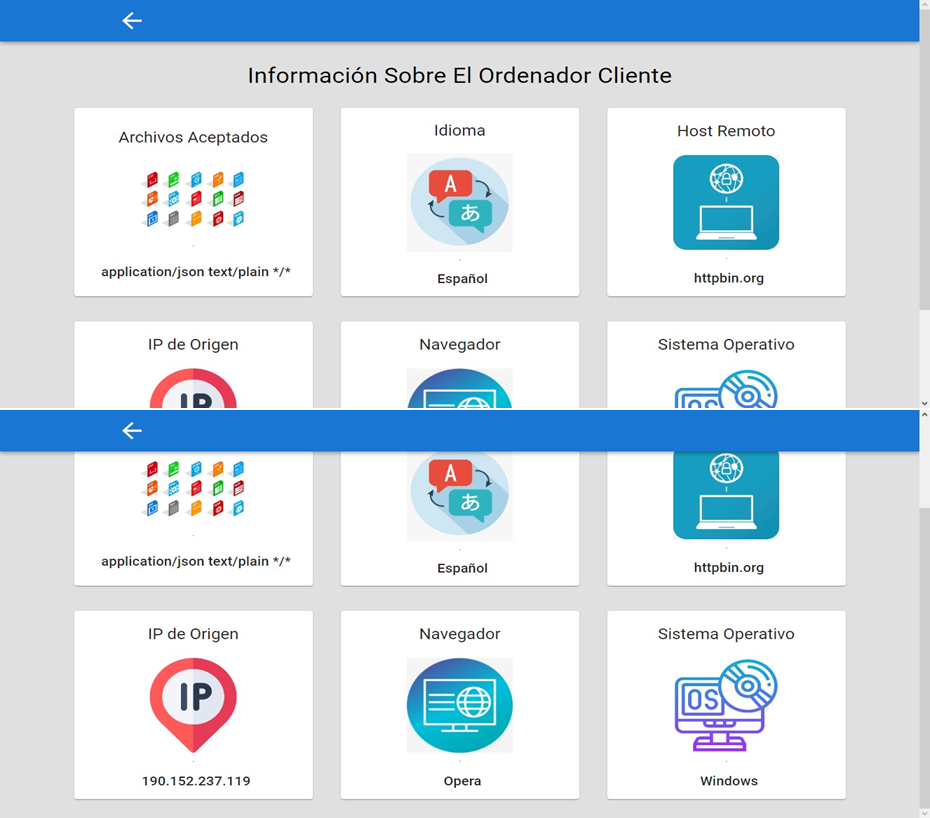
**You must also create a page that shows the information of the client computer from where you are accessing the API. For that, the public API that you are using provides the following URI https://httpbin.org/get. That URI returns the data of the client machine in JSON format. The data to be submitted are (in a web page, i.e., the data must be formatted as a web page, not JSON):**

* The files that are accepted
* The language
* The remote host
* The IP that originated the request
* The name of the browser
* The Operating System of the local machine

**2.4 Design:**

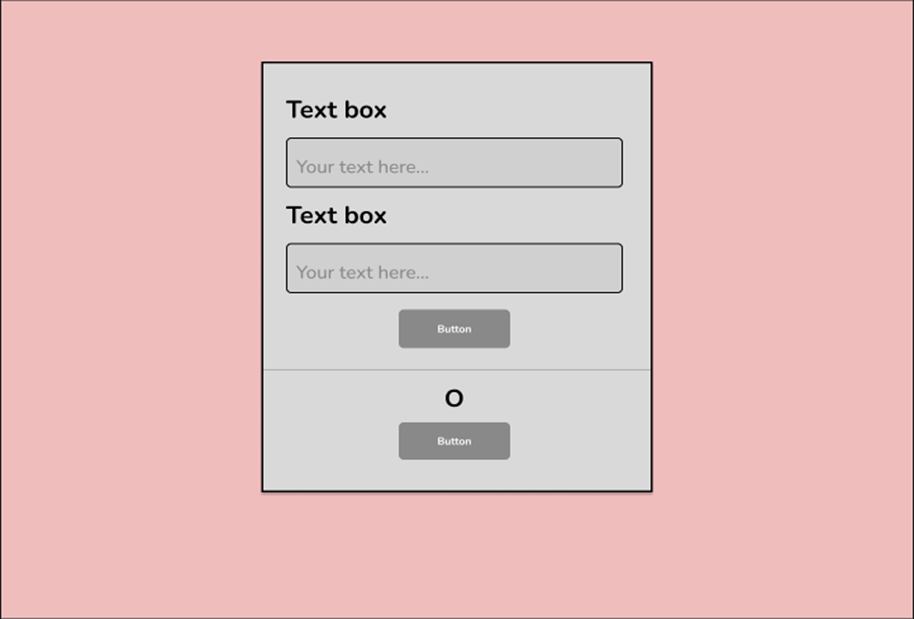
****

**2.4.1 Implementation:**

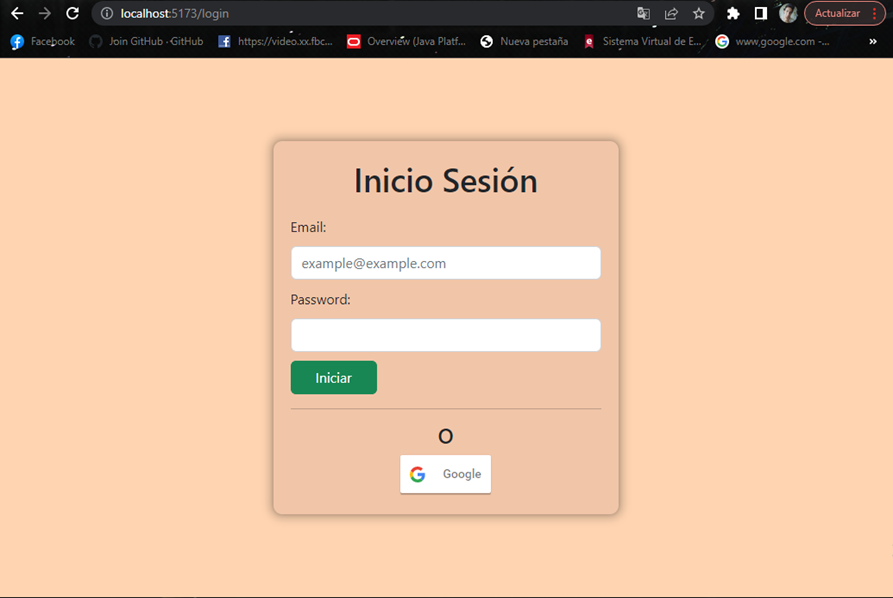


**USER INTERFACES FOR YOUR PROJECT**

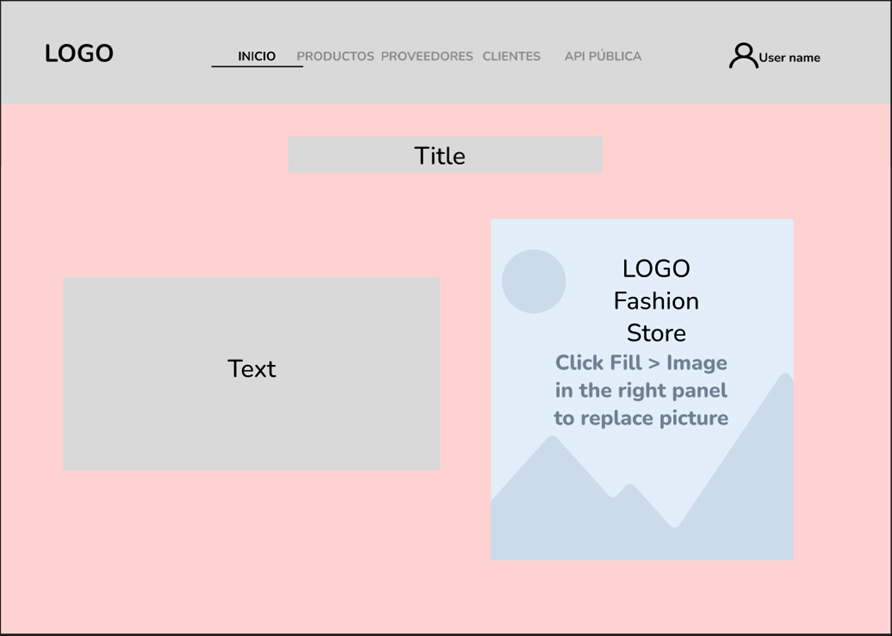
**2.5 Login Design**

****

**2.5.1 Implementation login**

****

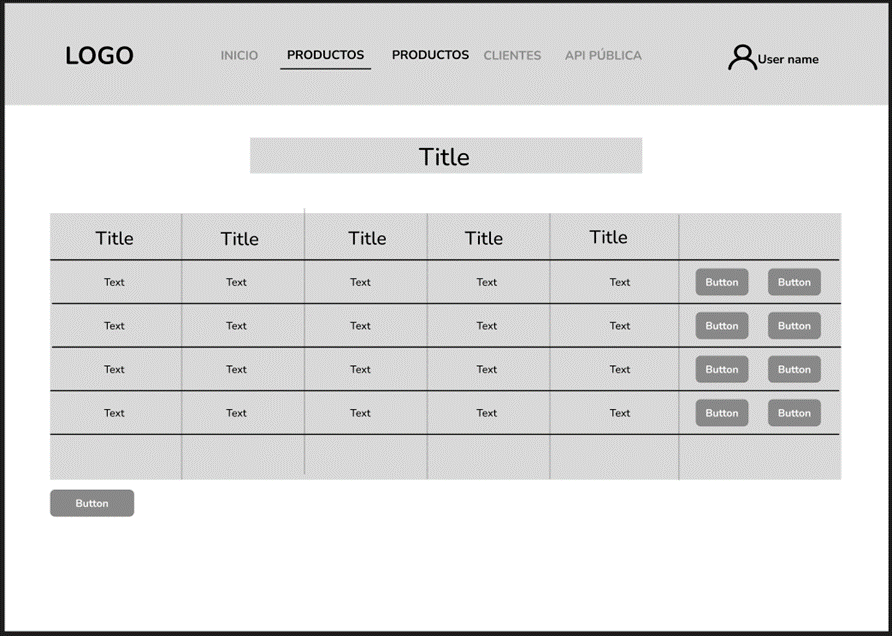
**2.6 Home Page Design**

****

**2.6.1 Implementation Home Page**

****

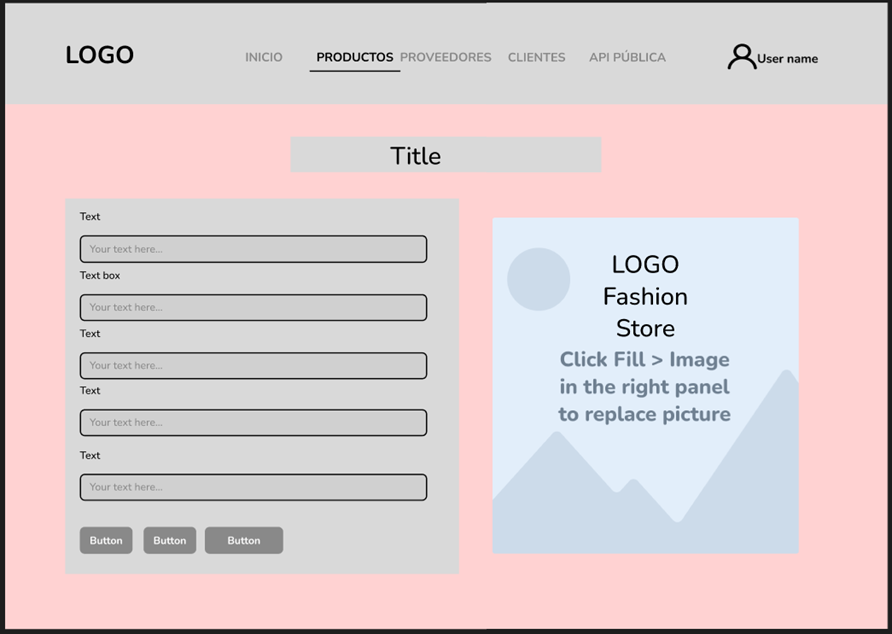
**2.7 Product List Design**

****

**2.7.1 Implementation List Products**

****

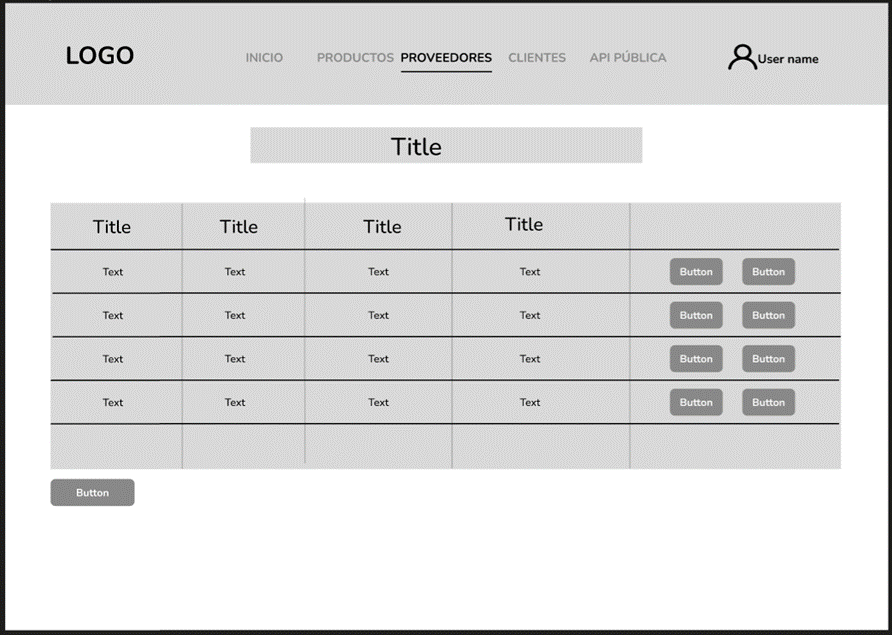
**2.8 Design Add Product**

****

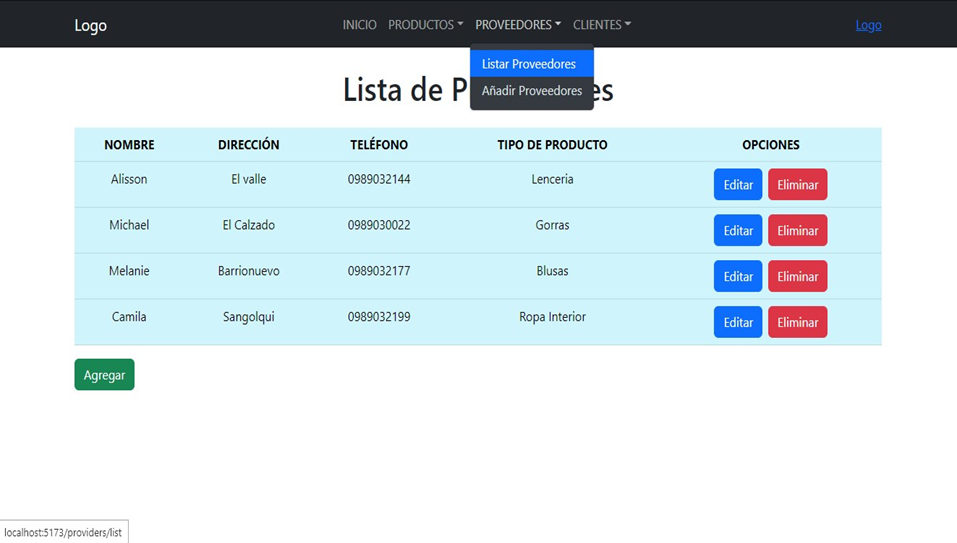
**2.8.1 Implementation Add Product**

****

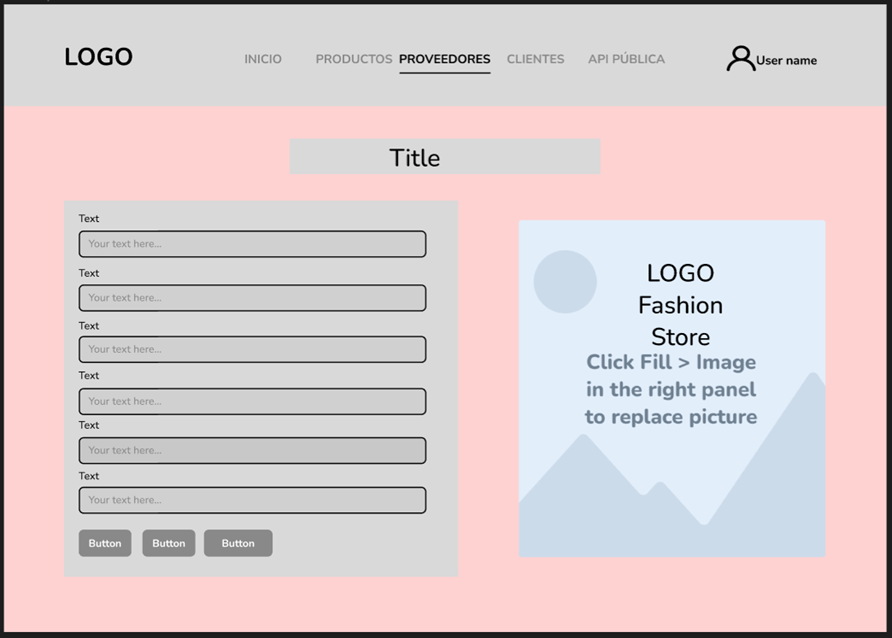
**2.9 Provider List Design**

****

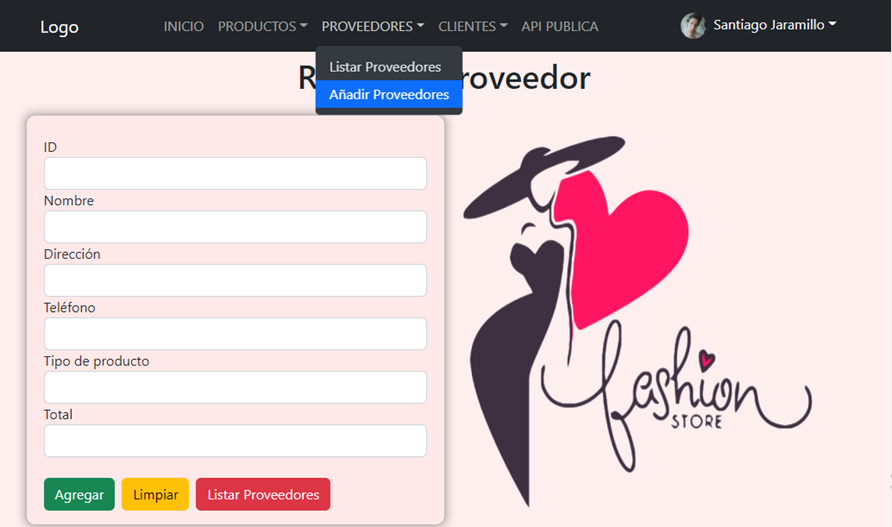
**2.9.1 Implementation List Provider**

****

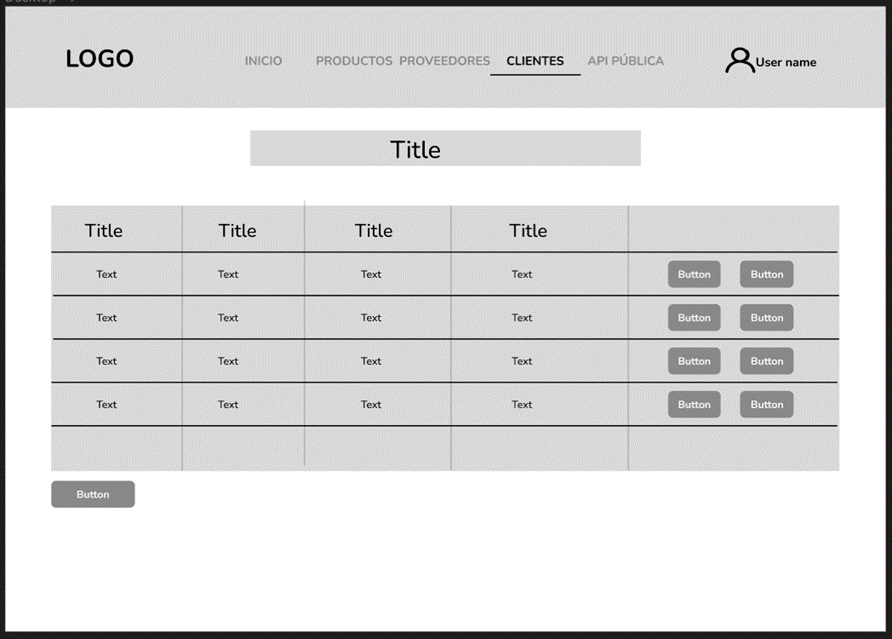
**2.10 Add Provider Design**

****

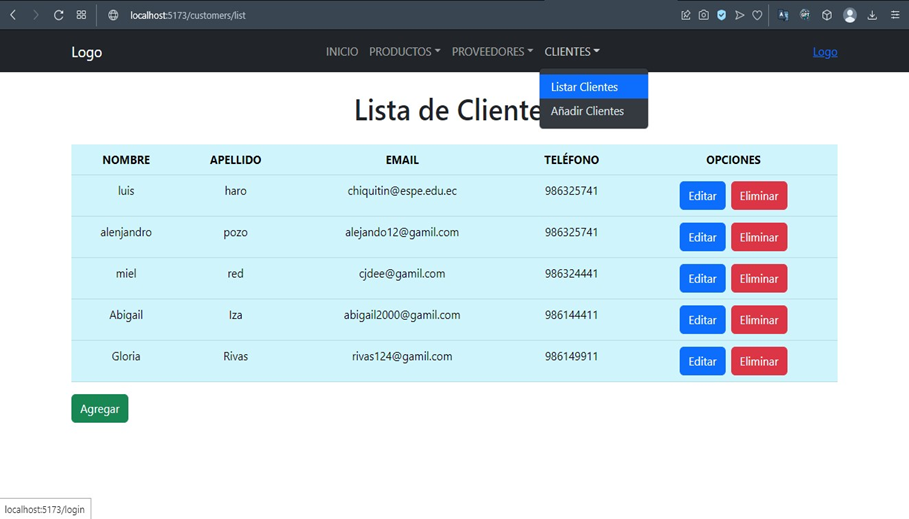
**2.10.1 Add Provider Implementation**

****

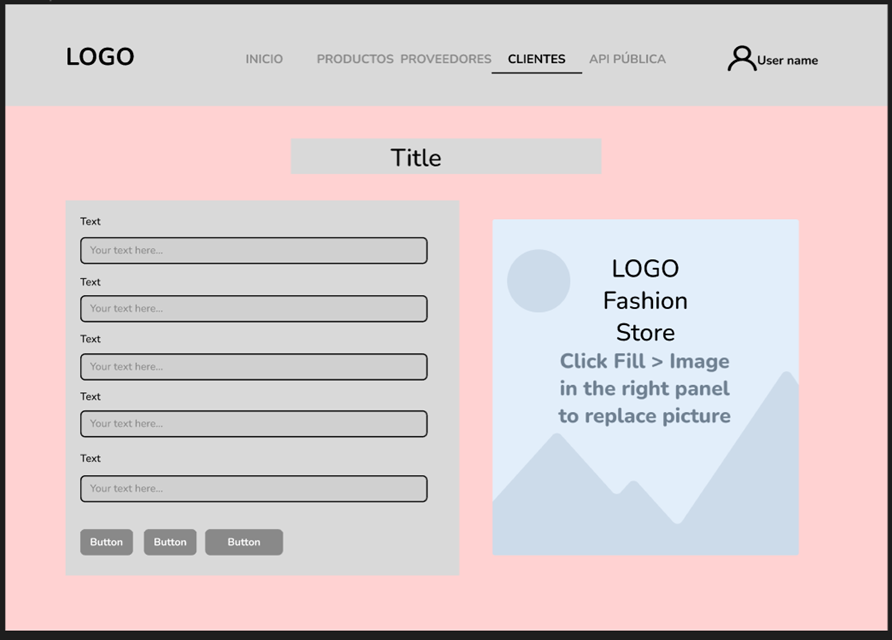
**2.11 Customer List Design**

****

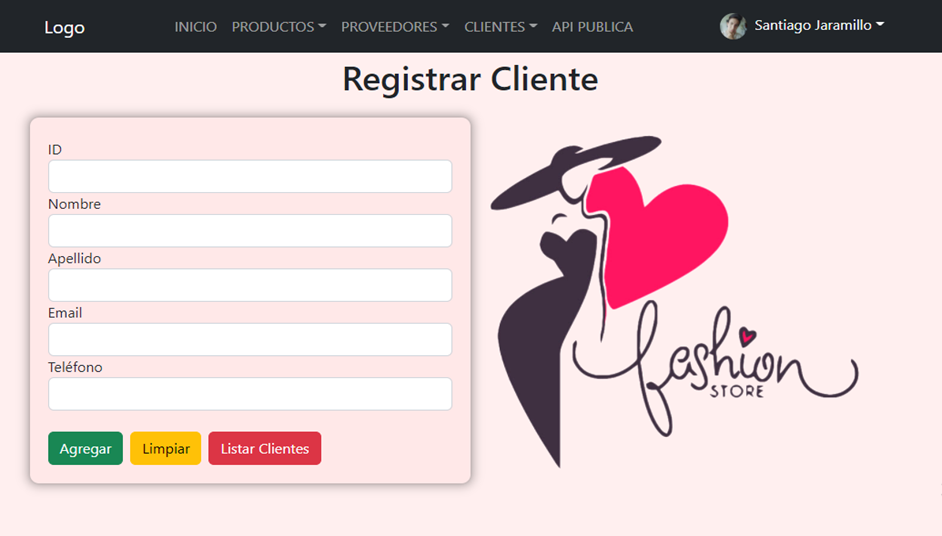
**2.11.1 Client List Implementation**

****

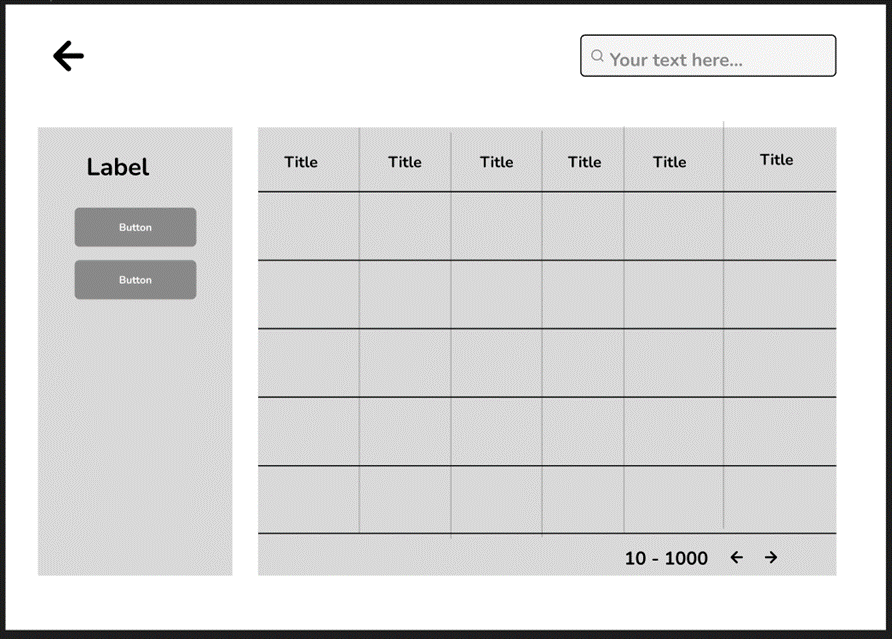
**2.12 Design Add Customer**

****

**2.12.1 Implementation Add Client**

****

**2.13 Design Consumer Api get product and users**

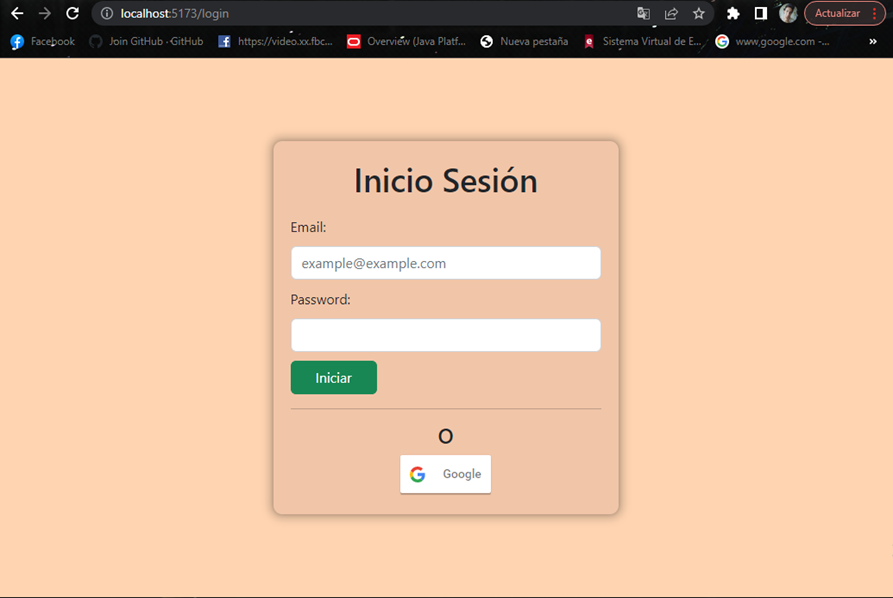
****

**2.13.1 Implementation of Api consumption get products and users**

****

**3. Authentication and Navigability (sessions) /0.5**

**Login**

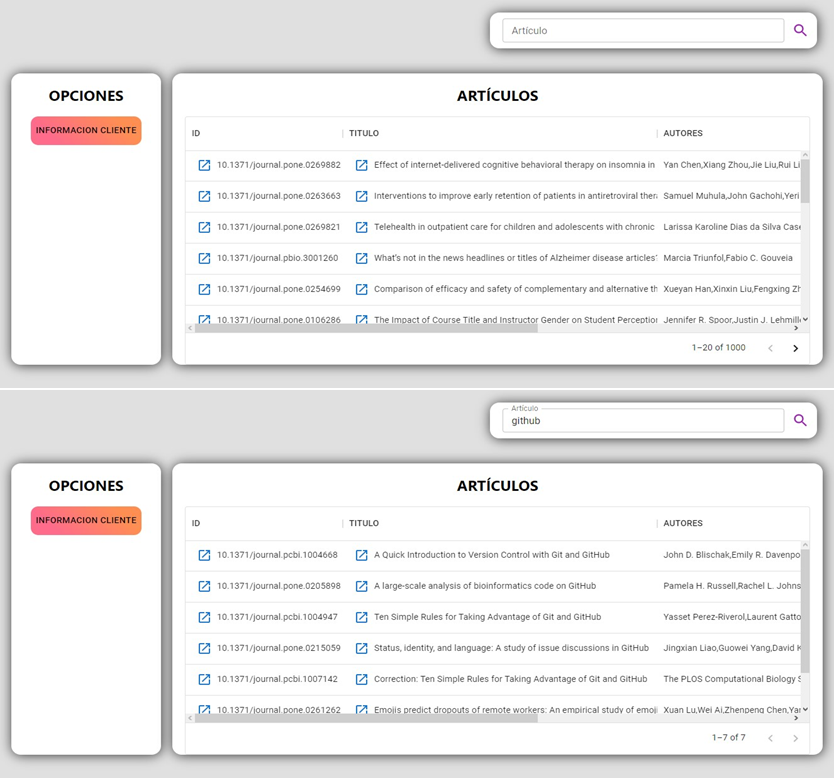
****

**Home**

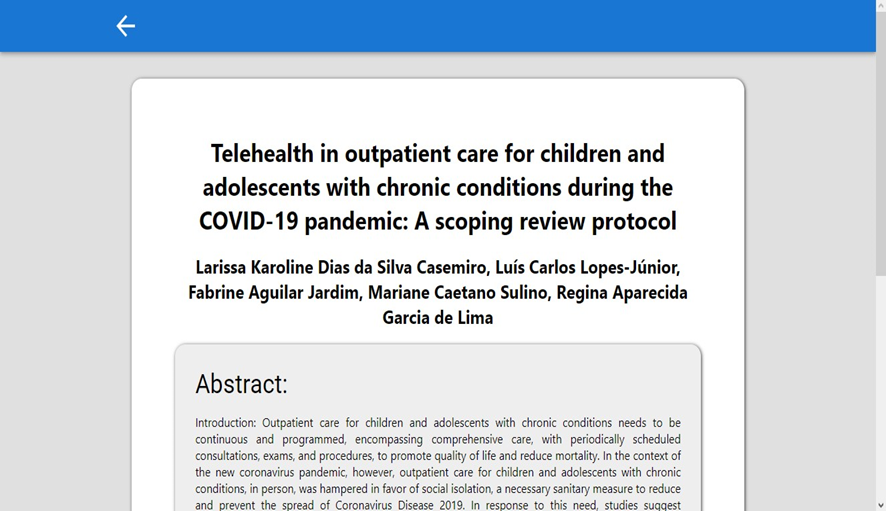
****

**4. Web Clients + public API https://api.plos.org/search /0.5**

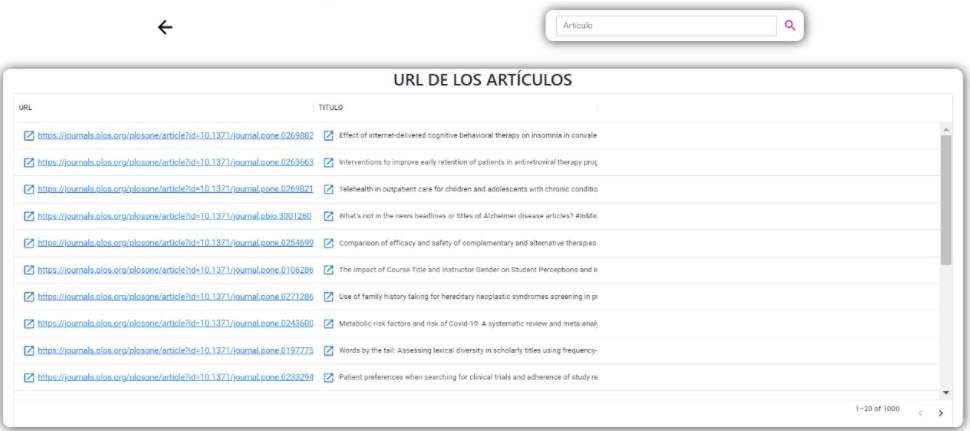
* **This page shows the tabulated documents**

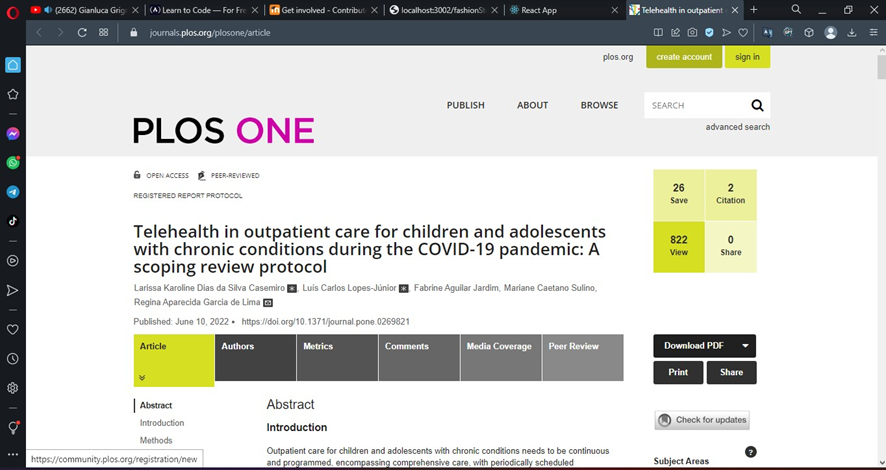
****

* **Program a link that displays the abstract for that paper on a new page as read from the URI**

****

* **A new page shows a table of every document identifier, with their respective id. From each id, a URL must be created so that the entire document can be viewed in a new tab**

****

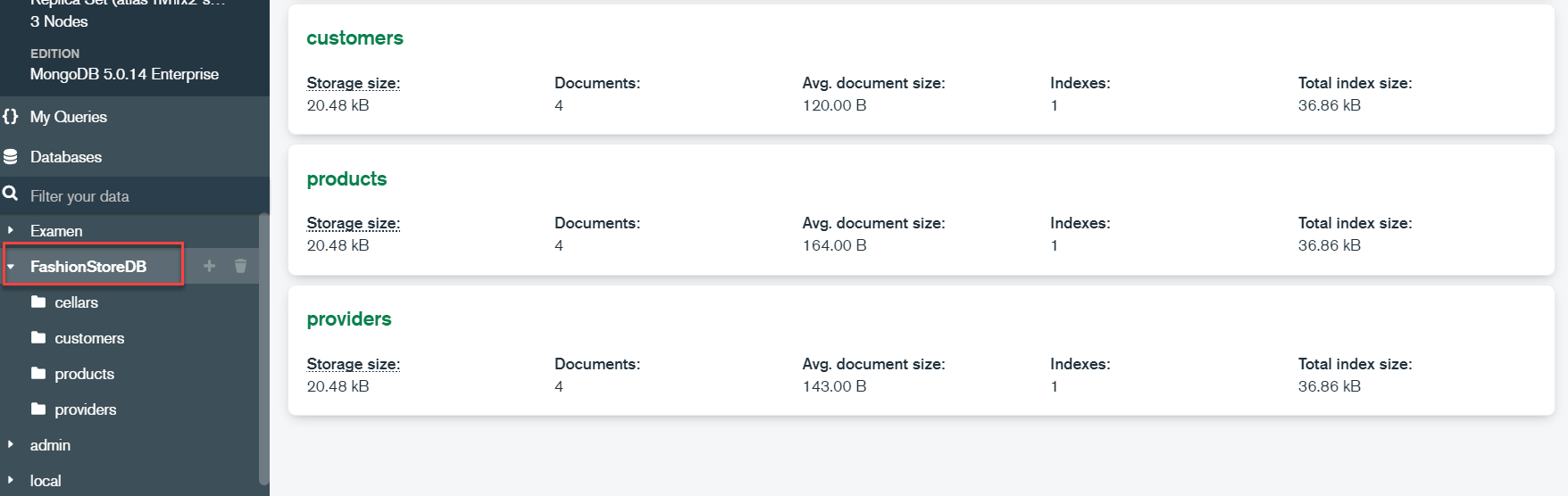
****

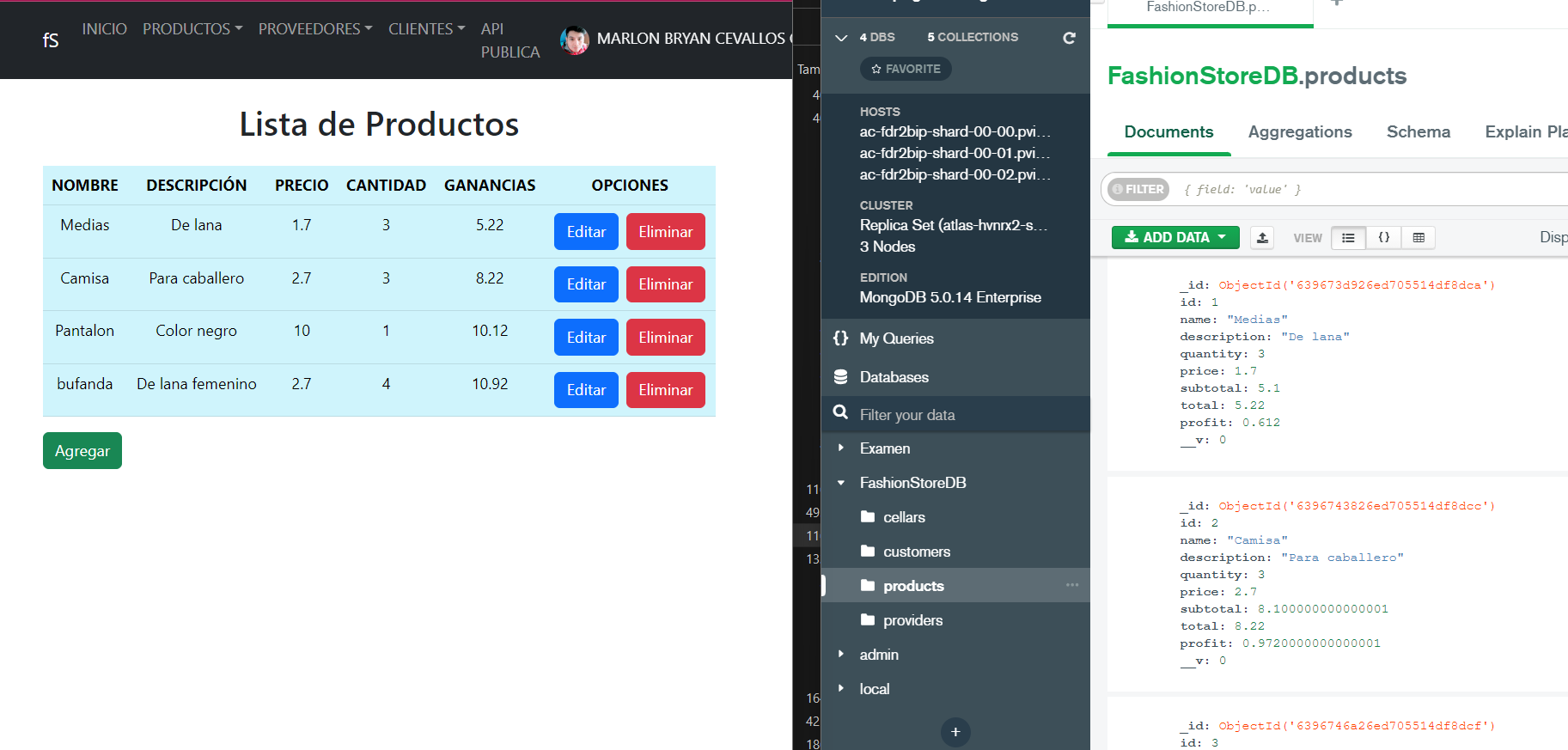
* **You must also create a page that shows the information of the client computer from where you are accessing the API. For that, the public API that you are using provides the following URI https://httpbin.org/get.**

****

**5. Web Clients + Web Service + BDD /0.5**

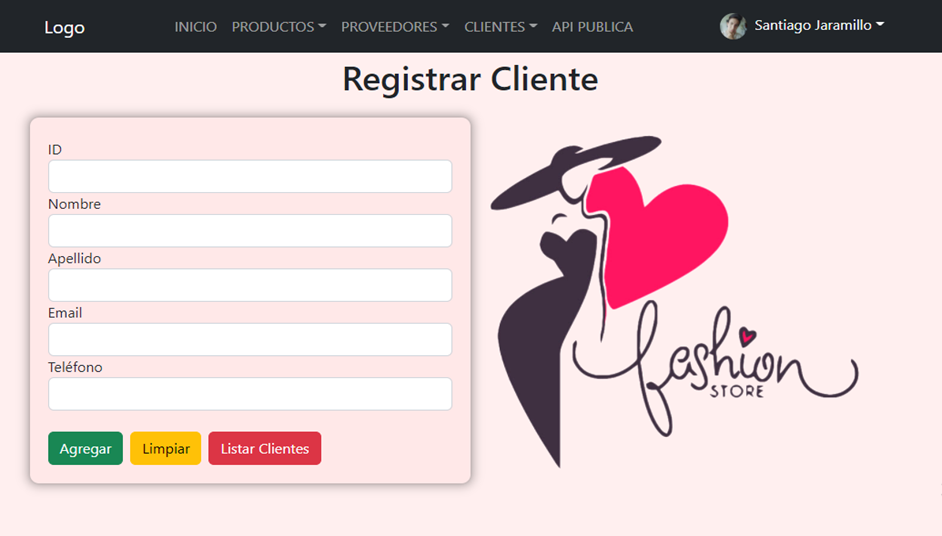
* Data Base and Web Service

****

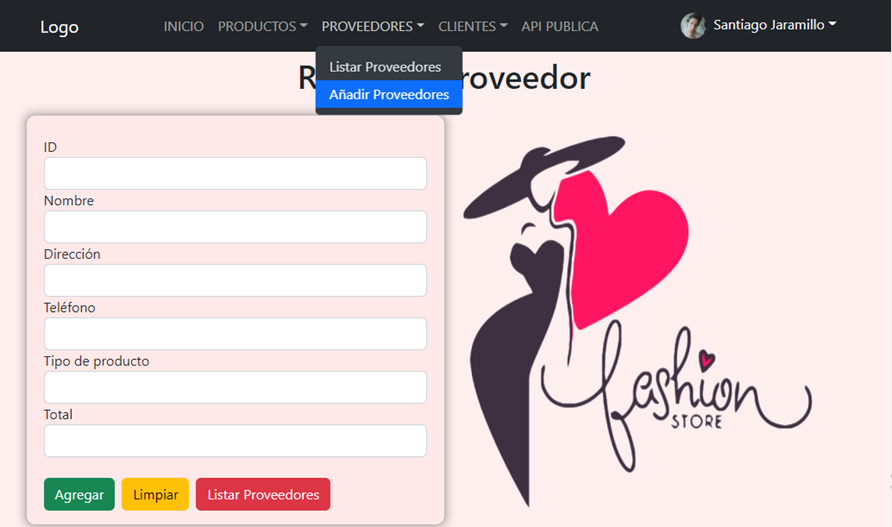
****

**6. Other web pages / forms for the project /1**

* **Form - register client**

****

* **Form - register provider**

****

* **Form - register product**

****